JC825 U.S. PTO
10/25/00
RECEIVED
MAY 3 1 2002
OPLA
JC474 U.S. PTO
10/042733
10/25/00

CONTINUING APPLICATION TRANSMITTAL

Address to:	Attorney Docket No.	18659-15C1
Assistant Commissioner for Patents	First Named Inventor	Priem, et al.
Box CPA	Express Mail Label No.	EL654882526US
Washington, DC 20231	Total Pages	2

This is a Continuing Application under 37 C.F.R. § 1.53(b) of, and claims the benefit of, prior application number, 09/056,386 filed on April 7, 1998 entitled "METHOD AND APPARATUS FOR PROVIDING A VERTEX CACHE"; and prior application number, 09/053,998 filed on April 2, 1998 entitled "METHOD AND APPARATUS FOR ACCELERATING THE RENDERING OF GRAPHICAL IMAGES.

1. ☐ Enter the unentered amendment previously filed on (fill in) under 37 C.F.R. § 1.116 in the prior nonprovisional application.

2. ☒ A Preliminary amendment and Request to Amend the Drawings are enclosed.

3. ☐ This application is filed by fewer than all the inventors named in the prior application, 37 C.F.R. § 1.53(d)(4).
   a. ☐ DELETE the following inventor(s) named in the prior non-provisional application:
   b. ☐ The inventor(s) to be deleted are set forth on a separate sheet attached hereto.

4. ☐ A new power of attorney or authorization of agent is enclosed.

5. ☒ A copy of original Declaration and Power of attorney is enclosed.

6. ☐ Information Disclosure Statement (IDS) is enclosed:
   a. ☐ PTO-1449
   b. ☐ Copies of IDS Citations

SV/110027.01
10252000/18:44/18659.15

FEE CALCULATION

	CLAIMS	(1) FOR	(2) NUMBER FILED	(3) NUMBER EXTRA	(4) RATE	(5) CALCULATIONS	
		TOTAL CLAIMS (37 C.F.R. § 1.16(c))	12 - 20	0	$ 18		
		INDEPENDENT CLAIMS (37 C.F.R. § 1.16(b))	3 - 3	0	$ 80		
		MULTIPLE DEPENDENT CLAIMS (if applicable) (37 C.F.R. § 1.16(d))		$260			
				BASIC FEE (37 C.F.R. § 1.16(a))		$710	
				Total of above Calculations =			
				Extension of Time Fee (1-month=$110; 2-month=$390; 3-month=$890)		$890	
		Reduction by 50% for filing by small entity (Note 37 C.F.R. §§ 1.9, 1.27, 1.28)					
					TOTAL =	$1,600	

6. ☐ Small Entity Status
    a. ☐ A small entity statement is enclosed.
    b. ☒ A small entity statement was filed in the prior nonprovisional application and such status is still proper and desired.
    c. ☐ Is no longer claimed.
7. ☐ The Commissioner is hereby authorized to charge Deposit Account 02-3964 the amount of $ _____.
8. ☒ We authorize the Commissioner to credit overpayments or charge deficiencies in the following fees to Deposit Account No. 02-3964:
    a. ☒ Fees required under 37 C.F.R. § 1.16.
    b. ☒ Fees required under 37 C.F.R. § 1.17.
    c. ☐ Fees required under 37 C.F.R. § 1.18.
9. ☒ We enclose a check in the amount of $710 for the filing fee.

10. ☒ We enclose a check in the amount of $890 for the extension fee.

☐ Other:

10. CORRESPONDENCE ADDRESS
Leah Sherry, Esq.
OPPENHEIMER WOLFF & DONNELLY LLP
1400 Page Mill Road
Palo Alto, California 94304
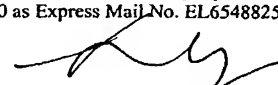Telephone (650)320-4000 • Fax (650) 320-4100

Date: <u>October 25, 2000</u>

Leah Sherry, Reg. No. 43,918

SV/110027.01
10252000/18:44/18659.15

IN THE PATENT AND TRADEMARK OFFICE

In Re Application of: Examiner: Unassigned

        Priem et al.   **RECEIVED**

Serial No.: Unassigned   Art Unit: MAY 3 1 2002

Filed: October 25, 2000

For: METHOD AND APPARATUS FOR PROVIDING A VERTEX CACHE   OPLA

Assistant Commissioner for Patents
Washington, D.C. 20231

PRELIMINARY AMENDMENT

Sir:

        In conjunction with the concurrent filing of a Continuing Application, and in
response to the Office Action dated April 25, 2000, the period for response to which is
extended, by the petition for extension of time, through October 25, 2000, please amend
the application and consider the remarks as follows:

In the Specification:

PREVIOUS AMENDMENTS

The specification as originally filed in the above-referenced Application is attached
herewith. However, in the amendment mailed March 17, 2000, the specification has been
amended as hereafter outlined and these amendments should now be incorporated into the
original specification as follows:

        On page 4, line 15, replace "Figure 2 is a" with --Figures 2A and 2B--; and delete
"of" after "diagram".

1

On page 8, line 20, insert immediately after "20" and before "which" --, as shown in Figures 2A and 2B,".

On page 9, line 25, replace "Figure 2," with --Figures 2A and 2B,--.

On page 10, line 6, delete "to" between "cache" and "by".

5   On page 14, line 4, insert immediately after "20 and before the "." --, as shown in Figure 9--.

NEW AMENDMENTS

Please further implement in the specification the following new amendments:

On page 1, line 2, before "BACKGROUND" insert a new paragraph as follows:

10   --**REFERENCE TO RELATED APPLICATION**

This Application is a continuing Application, and claims the benefit of U.S. Application No. 09/056,386, filed on April 7, 1998, and further claims the benefit of U.S. Application 09/053, 998, filed on April 2, 1998.--.

On page 4, line 8, before "These" insert a paragraph as follows:

15   --In one embodiment, the graphics accelerator may be adapted to produce screen values for the respective attributes using a texture engine, a scaling circuit, a setup circuit, and a lighting pipeline.--

On page 4, line 15, before replace "Figure 2" with:

--Figure 2A illustrates an exemplary graphics accelerator circuit which may be used in

20   accordance with the present invention with the computer of Figure 1.

Figure 2B --.

2

On page 9, line 25, replace "In Figure 2," with:

--Figure 2A illustrates an exemplary graphics accelerator circuit which may be used in accordance with the present invention with the computer of Figure 1. In general, the circuit 15 includes a setup engine 21 which receives input values for the various

5 attributes defining the three vertices of any triangular surface being depicted. Before transfer to the setup engine 21, the x, y, u, and v values representing the vertices of the triangle are processed by a scaling circuit 24. The scaling circuit 24 receives the x, y, u, and v values representing the vertices of the triangle and utilizes the maximum and minimum values of each to determine a proper scale for a texture map. The x, y, and z

10 coordinates may be provided by an application program as either screen space or world space values. The change from world space to screen space is based on a perspective transformation process. The scaling circuit 24 may also determine the pixels within a triangle from the x and y screen values of the vertices and generate specific screen x and y values for these pixels. The setup circuit 21 utilizes the x, y, and z screen values of the

15 vertices to determine screen values for each of the attributes of each pixel in the triangle. The process of computing perspective-correct screen values for the attributes from world space vertex values can expressed by a geometric relationship. Gating circuitry which carries out addition, subtraction, multiplication, and division steps can be employed to produce perspective correct screen values for each of the attributes at each pixel position.

20 The texture coordinates provided by the setup circuit 21 are transferred to a texture engine 22. The texture engine 22 utilizes those input values to determine texture coordinates at a plurality of positions within each pixel in accordance with the foregoing discussion of the invention. For example, the texture engine 22 may translate the texture coordinates at each of a plurality of positions within a pixel into texture values at those

25 positions and blend the texture values to realize a final texture value for each pixel. This final texture value is transferred to a lighting pipeline 23 where the texture value and the

3

other various attributes from the setup circuit 21 are utilized to modify the color value of the particular pixel in the sequence of pixels utilized to describe the triangle. From the lighting pipeline, the pixels are transferred to the rendering pipeline.

In Figure 2B,--.

1 **In the Claims:**

2 Please cancel claims 1-12 without prejudice and add new claims 13-24 as follows:

3 13.   A computer comprising

4   a central processing unit;

5   a bus;

6   memory;

7   a graphics accelerator coupled to the bus including a cache for data associated with

8 vertices, the data for each of the vertices representing its respective attributes including its screen

9 coordinates, color and a fog attribute, the data of any of the vertices that is already resident in the

10 cache can be used and reused to render any polygons defined by the vertices; and

11   a circuit producing screen values for the respective attributes, the circuit including a

12 texture engine, a scaling circuit, a setup circuit, and a lighting pipeline.

1 14.   A computer as in claim 13 in which the cache includes cache positions for the data

2 associated with the vertices the cache positions being indexed by an application program.

3

4 15.   A computer as in claim 13 in which the cache has a memory mapped storage space for the

5 data associated with the vertices of polygons.

1 16.   A computer as in claim 13 in which the cache includes cache positions for the data

2 associated with the vertices the cache positions being indexed by an address in a data structure of

3 vertices in memory.

4

1   17.   A computer as in claim 13 further comprising a direct memory access (DMA) engine for

2   transferring the data to the cache.

1   18.   A method for rendering polygons, comprising:

2   providing indices for designating cache positions of data in a cache, the data being

3   associated with vertices of any of the polygons and defining for each of the vertices its respective

4   attributes including its screen coordinates, color and a fog attribute;

5   storing the data at the designated cache positions in the cache based on each of the indices

6   being provided, the data for any of the vertices that is already resident in the cache can be used

7   and reused to render any polygons defined by the vertices;

8   issuing a command for rendering a polygon by indicating the indices for selecting the

9   data associated with the vertices of the rendered polygon; and

10   producing in a circuit screen values for attributes respective to the vertices of the rendered

11   polygon, the circuit including a texture engine, a scaling circuit, a setup circuit, and a lighting

12   pipeline.

1   19.   A method as in claim 18 in which the step of issuing the command further includes

2   directing a central processing unit to transfer the command to a register.

3   20.   A method as in claim 18 in which the step of issuing the command further includes

4   embedding the command in the data associated with the polygon.

1   21.   A method as in claim 18 in which the step of providing the indices further includes

2   creating an array of vertices in a memory,

3   indexing data for each of the vertices which is stored in the array,

4   selecting from the array vertices defining a polygon to be rendered, and

5   transferring to the cache the data for each of the selected vertices.

1   22.   A method as in claim 18 further comprising transferring the data to the cache by

2   commanding a direct memory access (DMA) engine to transfer the data from an array of vertices

3   in a memory.

1   23.   A graphics accelerator comprising:

2   setup circuitry responding to data defining attributes of vertices of polygons for

3   determining attributes of pixels to be displayed, the attributes of each of the vertices including

4   screen coordinates, color and a fog attribute, the setup circuit communicating with a texture

5   engine, a scaling circuit and a lighting pipeline; and

6   a cache circuit for storing the data, the data of any of the vertices that is already resident

7   in the cache circuit can be used and reused to render any polygons defined by the vertices.

8   24.   A graphics accelerator as in claim 23 further comprising a direct memory access (DMA)

9   engine for transferring the data to the cache circuit in response to commands from an application

10 program.

6

REMARKS

THE CLAIMS

Claims 1-12, have been canceled. New claims 13-24 have been added and are presently pending in the Application. .

5   THE SPECIFICATION

The specification has been amended to remove informalities therefrom and render the specification consistent with the drawings. The specification is now believed to be clear and concise.

THE DRAWINGS

10   A request to amend the drawings in accordance with 37 C.F.R. § 1.12(b)(3) is attached herewith. The request is seeking approval of Fig. 2A and a proposed change to the original Fig. 2. A red-marked copy of Fig. 2A and Fig. 2B are furnished herewith with the request.

CONCLUSION

15   In view of the foregoing amendments and remarks, it is respectfully submitted that the application is now in condition for allowance. Accordingly, a Notice of Allowance of claims 13-24, is respectfully requested.

If for any reason an insufficient fee has been paid, the Commissioner is hereby authorized to charge any deficiency in payment of required fees associated with this

20   communication to Deposit Account **02-3964.**

Date: October 25, 2000   Respectfully submitted,

Oppenheimer Wolff & Donnelly LLP
1400 Page Mill Road
Palo Alto, CA 94304   By: Leah Sherry,
                      Attorney for Applicant

7

IN THE PATENT AND TRADEMARK OFFICE

In Re Application of:	Examiner: unassigned
Priem et al.	
Serial No.: unassigned	Art Unit:
Filed: October 25, 2000	
For: METHOD AND APPARATUS FOR PROVIDING A VERTEX CACHE	

RECEIVED

MAY 3 1 2002

OPLA

Assistant Commissioner for Patents
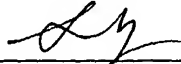Washington, D.C. 20231

REQUEST TO AMEND THE DRAWINGS

Sir:

In accordance with 37 C.F.R. § 1.12(b)(3), Applicants respectfully submit Fig. 2A and propose an amendment to original Fig. 2 of the drawings to render the drawing more clear and consistent with the specification. The request is seeking approval of a proposed change to original Fig. 2. A red-marked copy of Fig. 2A, and Fig. 2B are furnished herewith with the request. No new matter is introduced by the proposed amendment to the drawings.

Date: October 25, 2000

Oppenheimer Wolff & Donnelly LLP
1400 Page Mill Road
Palo Alto, CA 94304

Respectfully submitted,

By: Leah Sherry,
Attorney for Applicant
Reg. No. 43,918

1

Our Ref: NV29

APPLICATION FOR UNITED STATES LETTERS PATENT

FOR

METHOD AND APPARATUS FOR PROVIDING A VERTEX CACHE

Inventors:

Curtis Priem
David Kirk

Prepared by:

Stephen L. King
30 Sweetbay Road
Rancho Palos Verdes, CA 90275
(310) 377-5073

METHOD AND APPARATUS FOR PROVIDING A VERTEX CACHE

BACKGROUND OF THE INVENTION

Field Of The Invention

This invention relates to computer systems, and more particularly, to
methods and apparatus for providing a cache to hold data representing
vertices of polygons being displayed by a computer graphics output
device.

History Of The Prior Art

In three dimensional graphics, surfaces are typically rendered by
assembling a plurality of polygons in a desired shape. The polygons are
conventionally triangles having vertices which are defined in world space
by three dimensional coordinates, color values, texture coordinates, fog
values, and other values. The three dimensional world space coordinates
are translated into screen coordinates in which horizontal and vertical
values define screen position and a depth value determines how near a
vertex is to the screen and thus whether that vertex is viewed with
respect to other points at the same screen coordinates. The color values
define the brightness of each of red/green/blue colors at each vertex and
thus the color at each vertex. The texture coordinates fix each vertex on
a texture map, a matrix of values stored in memory which together
describe a pattern to be applied to the surface of the triangle to vary the
color values in accordance with the pattern.

The graphics accelerator utilizes the three dimensional coordinates
received from the central processing unit to define the vertices of a

triangle in screen space and from those to determine the individual pixels describing each triangle. For each pixel of the triangle, the graphics accelerator carries out a series of complex manipulations to determine the color values, depth values, texture coordinates, and other attributes

5   in two dimensional screen space. Once these attributes are determined for a pixel, the accelerator uses the texture coordinates to generate a texture value for each pixel in another complex manipulation. Finally, all of the screen attributes of a pixel which affect the color of that pixel are combined to provide final color values for the pixel; and these pixel data

10   are placed with the pixel address and depth in a rendering pipeline.

As may be appreciated, the amount of data required to define each vertex in an advanced three dimensional system is substantial. In one particular arrangement, thirty-two bytes of data are required to describe a single vertex. To accomplish the operations by which the individual

15   pixels which describe the triangle are defined for use by some graphics output device, it is first necessary to transfer the data defining each vertex of each triangle from system memory to the graphics accelerator circuitry. Conventionally, data defining each individual vertex of a triangle is individually transferred by the central processing unit to the

20   graphics accelerator over the system input/output (I/O) bus. This requires that the central processing unit use the bus control circuitry to gain access to the system (I/O) bus in order to transfer the data defining each of the vertices.

Twenty to forty bytes of data are typically required to define all of the

25   attributes at each of the vertices in world space. In a computer with a thirty-two bit bus I/O bus, five to ten writes by the central processing

unit are needed to transfer data describing a single vertex. On the other hand, a command may require less than a byte. Consequently, to transfer data defining three vertices and a command to render a triangle requires between sixteen and thirty-one bus transfers. If each transfer requires a bus acquisition, it may take from sixteen to thirty-one bus acquisitions to transfer the data describing a single triangle. Even when burst transfers are utilized, no more than a single vertex can be transferred in one burst so as many as four bus acquisitions are required to transfer the vertex data and a command to render a single triangle. Thus, both normal and burst transfers are relatively slow processes constrained by bus bandwidth and require a significant amount of central processor time.

It is desirable to transfer data for rendering polygons more rapidly between a source of graphics data and graphics accelerator circuitry in a manner which reduces the effect of bus bandwidth.

Summary Of The Invention

This and other desirable results of the present invention are realized by apparatus and a method which includes the steps of transferring data defining individual vertices of a polygon from a source of graphics data to a graphics accelerator, caching data defining each of the vertices in a cache until all vertices of a complete polygon are described, transferring a command to the graphics accelerator to render a polygon defined in terms of vertices in the cache, and rendering the polygon defined by the data for the vertices.

NV29

Once data defining all vertices of a complete polygon are in the cache, any polygon using those vertices may be rendered by simply transferring a command to render the polygon defined in terms of its vertices. Thus, if sufficient vertices are in the cache, a series of rendering commands may render a series of additional polygons without the need to transfer any additional vertices by simply reusing stored vertices to form different polygons.

These and other objects and features of the invention will be better understood by reference to the detailed description which follows taken together with the drawings in which like elements are referred to by like designations throughout the several views.

Brief Description Of The Drawings

Figure 1 is a block diagram describing a computer utilizing the present invention.

Figure 2 is a diagram of a cache memory which may be used in accordance with the present invention with the computer of Figure 1.

Figure 3 is an illustration of data defining three vertices of a polygon to be rendered by a graphics accelerator and a command to render a polygon used in accordance with an embodiment of the invention.

Figure 4 is an illustration of data defining a vertex of a polygon to be rendered by a graphics accelerator and a command to render a polygon used in accordance with another embodiment of the present invention.

Figure 5 is yet another illustration of data defining a vertex of a polygon to be rendered by a graphics accelerator in accordance with the present invention.

Figure 6 is a diagram illustrating a first arrangement of polygons defining a shape which may be rendered by a graphics accelerator in accordance with the present invention.

Figure 7 is a diagram illustrating a second arrangement of polygons defining another shape which may be rendered by a graphics accelerator in accordance with the present invention.

Figure 8 is a diagram illustrating a third arrangement of polygons defining another shape which may be rendered by a graphics accelerator in accordance with the present invention.

Figure 9 is a diagram illustrating the use of a hardware controlled cache designed in accordance with the present invention.

Detailed Description

Figure 1 is a block diagram illustrating a computer 10. The computer 10 includes a central processing unit 11, main memory 12, system input/output (I/O) bus 13, and various I/O devices 14 and 15. The I/O device 15 is a graphics accelerator designed in accordance with the present invention to speed the transfer of graphics data from the central processing unit 11 to the device 15. The graphics accelerator includes circuitry for rasterizing the data transferred, applying texture information, and rendering the pixel data defining the triangle to a frame

buffer. In addition, the device 15 includes a DMA engine 17 and a cache 20 designed in accordance with the present invention.

An application program drawing three dimensional shapes typically renders surfaces by assembling a plurality of polygons in a desired shape. The polygons are conventionally triangles which are defined by three dimensional coordinates in world space, color values, and texture coordinates of their vertices. Figure 6, for example, is a diagram illustrating the representation of a circle by a series of triangles having a number of vertices A-H and J. As may be seen, the triangles which represent the circle have a number of vertices which are identical from triangle to triangle. The vertex J, for instance, appears in each of the triangles defining the circle.

Figure 7 illustrates a rectangular shape defined by a series of triangles referred to as a "strip." In the strip of Figure 7, each vertex (other than those at the corners of the rectangle) such as L is shared by three different triangles. Figure 8 illustrates a larger rectangular shape defined by an arrangement of triangles referred to as a "mesh." As may be seen, internal vertices such as M in the mesh configuration are shared by six different triangles. In fact, in a large mesh, the average number of triangles which use each vertex is six.

Conventionally, an application program executing on the central processing unit 11 renders each triangle of a shape by transferring data defining the three vertices of the individual triangle followed by a rendering command for that triangle from system memory to the graphics accelerator 15. This requires that the central processing unit

6   NV29

11 gain access to the bus as described above in order to transfer the vertex data for each triangle in the shape.

Figure 3 is an illustration of the data required to be transferred over the system bus to the graphics accelerator in order to define three exemplary

5   vertices A, B, and J and a command to render a triangle ABJ in forming the circle of Figure 6. In Figure 3, each box represents four bytes of data. As may be seen, four bytes are required to represent each attribute x, y, z, u, and v while less than four bytes are required to represent each attribute r, g, and b. Similarly, representing the fog attribute (f) requires

10   less than four bytes. In contrast to the amount of data required to represent a vertex, the command to render the triangle ABJ requires, in one embodiment, only four bits of data. Because of the amounts of data to be transferred, the transfer operation for pixel data defining one vertex of a triangle typically requires as many as ten bus accesses to

15   accomplish.

It should be noted that the command format allows a "no-op" command to be implemented by naming all three vertices identically (e.g., JJJ); since such a command describes a point, no triangle is drawn, and the effect is a "no-op" command.

20   Thus, to render each triangle, the central processing unit 11 has to acquire access to the system I/O bus 13 as few as ten and as many as thirty-one times to transfer the vertex data and a rendering command to the graphics accelerator 15. Historically, this has not slowed operations appreciably since graphics accelerators have not functioned very rapidly.

25   However, as graphics accelerators have become faster, central processing

units have been unable because of the limited bus bandwidth to supply sufficient data to a graphics accelerator to keep the accelerator operating at full speed.

The operation of the computer may be accelerated by a new process by which an application program establishes a large transfer buffer in memory in which data describing a very large number of vertices may be stored. The application program commands a direct memory access (DMA) engine which may be positioned with the graphics accelerator 15 to transfer from the transfer buffer to the graphics accelerator the vertex data needed to render a polygon. The application program commands the graphics accelerator to render the triangle utilizing the vertex data transferred by the DMA engine. Even though the use of a DMA engine reduces the use of the central processing unit and thereby accelerates operations of the computer as a whole, the same bus bandwidth restrictions apply to transfers by the DMA engine so the transfer process itself is not faster.

The present invention reduces the need for bus bandwidth and allows triangles to be rendered at a speed at which the graphics accelerator is designed to operate. In order to accomplish this, the present invention utilizes a cache 20 which is a part of the graphics accelerating device 15 to store vertices. The cache 20 stores vertices of polygons as they are transferred to the device 15 either by a DMA engine or the central processing unit. The vertices are then held in the cache 20 until replaced. Once at least three vertices of a first triangle has been transferred to the graphics accelerating device 15 and stored in the cache 20, any vertex data in the cache may be used and reused to render any

triangles defined by those vertices. Since a typical shape rendered on the output display (e.g., the shape illustrated in Figure 6) is assembled from a plurality of triangles many of which have vertices which are vertices of other triangles, if vertices of one triangle are in the cache and a first triangle has been rendered using these vertices, it requires only the addition of data defining a single additional vertex to allow the rendering of a second triangle. Thus, a first triangle (for example, triangle ABJ in Figure 6) may be rendered by the transfer of the three vertices A, B, and J and a rendering command. Then, a triangle adjoining the first triangle ABJ and having two of the vertices of the first triangle (such as the triangle BCJ) may be rendered by the transfer of a single additional vertex (C) to the cache 20 followed by a command to the graphics accelerator 15 to render the triangle BCJ. Thus, rather than requiring three vertices and a render command to render an additional triangle after the first triangle of an interrelated group of triangles has been rendered, only a single additional vertex and a single render command need be transferred over the system bus. Since the amount of data defining a vertex varies from twenty to forty bytes, this may reduce the amount of data to be transferred in order to generate a second triangle by up to eighty bytes. This is a significant reduction in the data which needs to be transferred in order to render any triangle which utilizes vertex data already in the cache 20. In this simple case, the process divides the time required to transfer data across the system bus almost in half.

In Figure 2, an embodiment of the cache 20 is illustrated in which data defining each vertex may be stored by the application program executing

on the central processing unit 11. A particular cache 20 includes positions for data defining sixteen individual vertices in one embodiment and may include space for a tag. A position in a cache storing sixteen vertices may be selected by the use of only four tag bits. By selecting the particular cache positions in which the vertices reside, the application may utilize vertex data already stored in the cache to by simply transferring commands designating the vertices to be used in any triangle to be rendered.

The vertices may be stored in the cache 20 in a number of different ways. For example, the storage space in the cache may be memory mapped so that an application directs data describing a particular vertex to a particular memory address and recalls the data from that memory address when it is to be used in describing a triangle. The data describing a particular vertex may be placed in the cache 20 under software control. This allows a cache position to be designated by an index which requires significantly less data to describe than does an address. Thus, the application program may select cache positions for storage of vertex data which may be designated in the manner in which the vertices are numbered in Figure 6, for example. A command to render a triangle may designate a triangle by these same indexed representations and thereby require the transfer of less command data over the bus.

Software control over the storage of the data in the cache 20 may be accomplished by an application program which provides the means to generate a list of indexed positions in the cache 20 and also provides a replacement policy for the data in the cache 20. This allows the

NV29

application to designate the vertices making up a triangle to be rendered in commands in accordance with an indexing designation used for the storage space. In a typical case, such a designation requires four bits of data space to implement a sixteen entry cache. In such an embodiment

5   of the invention, the central processing unit store vertices in the cache 20 in storage spaces which the application program designates. Then, the central processing unit designates a triangle to be rendered by simply designating the indices of three vertices already stored in the cache 20 using a single command to the graphics accelerator which directs that

10   the rendering process commence.

Software control over the storage of the data in the cache 20 may also be accomplished in a similar manner by a software driver for the graphics accelerator which is equipped to generate a list of cache positions and to execute a replacement policy for data in the cache.

15   The command data itself may be transferred from an application program to the graphics accelerator in a number of ways. An application may cause the central processing unit to transfer a command via the system I/O bus to the graphics accelerator for execution after vertices have been placed in the cache. Such a command may be addressed to a particular

20   register on the graphics accelerator so that the receipt of vertex identifications at the register indicates the command intended. In a system utilizing a DMA engine to transfer vertex data, a command may be included within the data defining a particular vertex. For example, the data needed to represent fog (f) and the data needed to represent

25   r/g/b values each requires less than the four bytes allocated to it in the vertex data stream. The space not used by the fog attribute or the r/g/b

attributes in data defining a vertex may be utilized to transfer a command to render a triangle for which all of the vertices are in the cache or are included in the particular transfer of vertex data which includes the command to render. Such an arrangement of data is illustrated in Figure 4. This reduces the need to transfer commands separately and eliminates any bus access to assist the transfer of such a command. Once any two vertices of linking triangles have been transferred to the cache 20 using such a transfer method, it is only necessary to transfer data defining one additional vertex in order to commence the rendering of that triangle since the command for rendering may be included with the data defining the last vertex . In fact, it is possible to embed commands to render more than one triangle in the data defining a vertex and thereby lower the amount of data transferred even more.

Another manner of transferring commands to the graphics accelerator by a DMA engine is to add some data space to the format in which the data defining a vertex is transferred. For example, Figure 5 illustrates a format in which an additional four bit space is added to the left of the vertex definition data. Similarly, command data might be added at the right or any other position in the format. Using such a format requires more data transfer time than does the last described embodiment but is quite simple and easy to implement.

Another modification of the invention allows a series of triangles to be generated once vertices have been transferred for those triangles to a cache 20 in the various manners described above. For example, when the shape to be described is a strip such as is shown in Figure 7, all of

NV29

the vertices except those at the corners of the rectangle are used and reused in three different triangles. If only sixteen vertices have been transferred to the cache 20, the entire shape shown in Figure 7 may be described by sending a series of fourteen commands to render the fourteen triangles which may be defined by the sixteen vertices. A sequence of twelve bits of command data is sufficient to transfer a command to render any single one of the fourteen triangles since four bits will identify each vertex in the cache 20. Those skilled in the art will appreciate the reduction in data transfer for both vertices and commands which this allows. An even greater advantage is obtained in describing a shape such as the mesh of Figure 8. If fifteen vertices are placed in the cache 20 and a total of sixteen rendering commands are transferred, sixteen triangles are described for an output device.

Similar techniques as those described above for transferring commands may also be utilized for transferring data indicating the particular vertex which is being designated by a sequence of data being transferred. That is, data transferred by an application to a particular register may designate data to be stored in a particular cache position. In a data transfer using a DMA engine, the particular vertex may be designated by data indicated by an index value added to the format or included within excess space provided for attributes which do not use the entire space allotted by the format.

One considerable advantage of the present invention is that the cache 20 may be utilized to draw other than the typical triangles used to describe shapes. By defining a number of vertices, the shape of any particular quadrilateral may be defined. Thus, polylines (lines with fill of various

widths) may be drawn using the cache 20 and describing vertices which describe segments of the line.

An additional embodiment of the invention utilizes a hardware arrangement to manipulate the data stored in the cache 20. In order to accomplish such an arrangement, vertex data defining a large plurality of vertices to be displayed in stored in a portion of memory by the application program. For example, an application might place vertex data at any of the memory mapped positions in a memory array of 64K vertices. Each entry in the cache 20 carries a sixteen bit tag. A command initiated by an application includes three vertices each described by sixteen bits which are sufficient to access any vertex stored in the 64K region of memory. The command may be transferred to the graphics accelerator either by the central processing unit or a DMA engine on the graphics accelerator in the manner explained above. When the command is received by the graphics accelerator, the hardware cache controller of the accelerator checks the cache 20 to determine whether all of the vertex data exists in the cache 20. If the vertex data exists in the cache, the triangle may be rendered immediately. If any of the vertex data is not in the cache 20, the accelerator causes the DMA engine to read the vertex data in the region maintained by the application in memory and write the data to the cache 20. With the vertex data in the cache, the triangle is rendered. It should be noted that in such an arrangement, vertex data is not written across the system bus by the central processing unit. The vertex data is always read from memory by the DMA engine on the graphics accelerator. In order to function with the cache 20, the hardware control for the cache 20 may utilize a state

14   NV29

machine to implement an algorithm for replacing data in the cache where new data is required to draw triangles.

Although the present invention has been described in terms of a preferred embodiment, it will be appreciated that various modifications and alterations might be made by those skilled in the art without departing from the spirit and scope of the invention. The invention should therefore be measured in terms of the claims which follow.

<u>What Is Claimed Is</u>:

NV29

1 1.   (Amended) A computer comprising

2   a central processing unit;

3   a bus;

4   memory; and

5   a graphics accelerator coupled to the bus and including a cache for data associated with

6 vertices, the data for each of the vertices representing its respective attributes including its screen

7 coordinates, color and a fog attribute, the data of any of the vertices that is already resident in the

8 cache can be used and reused to render any polygons defined by the vertices.

1 2.   (Amended) A computer as claimed in Claim 1 in which the cache includes cache

2 positions for the data associated with the vertices the cache positions being indexed by an

3 application program.

1 3.   (Amended) A computer as claimed in Claim 1 in which the cache has a memory mapped

2 storage space for the data associated with the vertices of polygons.

1 4.   (Amended) A computer as claimed in Claim 1 in which the cache includes cache

2 positions for the data associated with the vertices the cache positions being indexed by an

3 address in a data structure of vertices in memory.

16

1   5.   (Amended)   A computer as claimed in Claim 1 further comprising a direct memory

2   access (DMA) engine for transferring the data to the cache.

1   6.   (Amended) A method for rendering polygons comprising:

2   providing indices for designating cache positions of data in a cache, the data being

3   associated with vertices of any of the polygons, for each of the vertices the data defines its

4   respective attributes including its screen coordinates, color and a fog attribute;

5   storing the data at the designated cache positions in the cache based on each of the

6   indices being provided, the data for any of the vertices that is already resident in the cache can be

7   used and reused to render any polygons defined by the vertices; and

8   issuing a command for rendering a polygon by indicating the indices for selecting the

9   data associated with the vertices of the selected polygon.

1   7.   (Amended) A method as claimed in Claim 6 in which the step of issuing the command

2   further includes directing a central processing unit to transfer the command to a register.

1   8.   (Amended) A method as claimed in Claim 6 in which the step of issuing the command

2   further includes embedding the command in the data associated with the polygon.

1   9.   (Amended) A method as claimed in Claim 6 in which the step of providing the indices

2   further includes

17

3   creating an array of vertices in a memory,

4   indexing data for each of the vertices which is stored in the array,

5   selecting from the array vertices defining a polygon to be rendered, and

6   transferring to the cache the data for each of the selected vertices.

1   10.   (Amended) A method as claimed in Claim 6 further comprising transferring the data to

2   the cache by commanding a direct memory access (DMA) engine to transfer the data from an

3   array of vertices in a memory.

1   11.   (Amended) A graphics accelerator comprising:

2   setup circuitry responding to data defining attributes of vertices of polygons for

3   determining attributes of pixels to be displayed, the attributes of each of the vertices including

4   screen coordinates, color and a fog attribute, and

5   a cache circuit for storing the data, the data of any of the vertices that is already resident in the

6   cache circuit can be used and reused to render any polygons defined by the vertices.

1   12.   (Amended) A graphics accelerator as claimed in Claim 11 further comprising a direct

2   memory access (DMA) engine for transferring the data to the cache circuit in response to

3   commands from an application program.

18

Abstract of the Disclosure:

A method for caching data defining vertices of a polygon to be displayed by an input/output display device including the steps of providing an index for a vertex for which data is to be cached, storing data defining

5   attributes of a polygon at a vertex in a cache under the index provided, issuing a command signifying a polygon to be manipulated by indicating indices of vertices of the polygon for which data is cached.
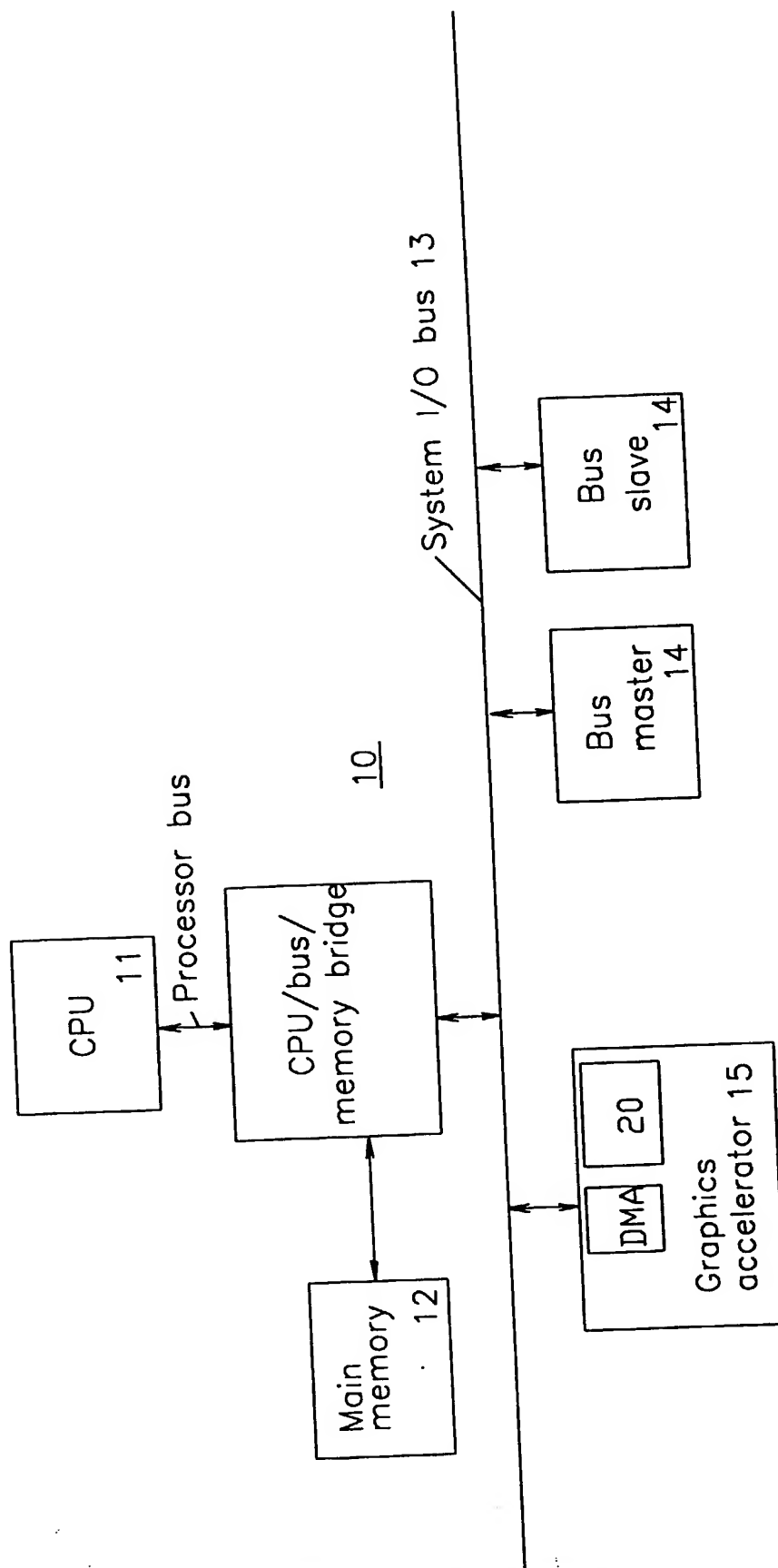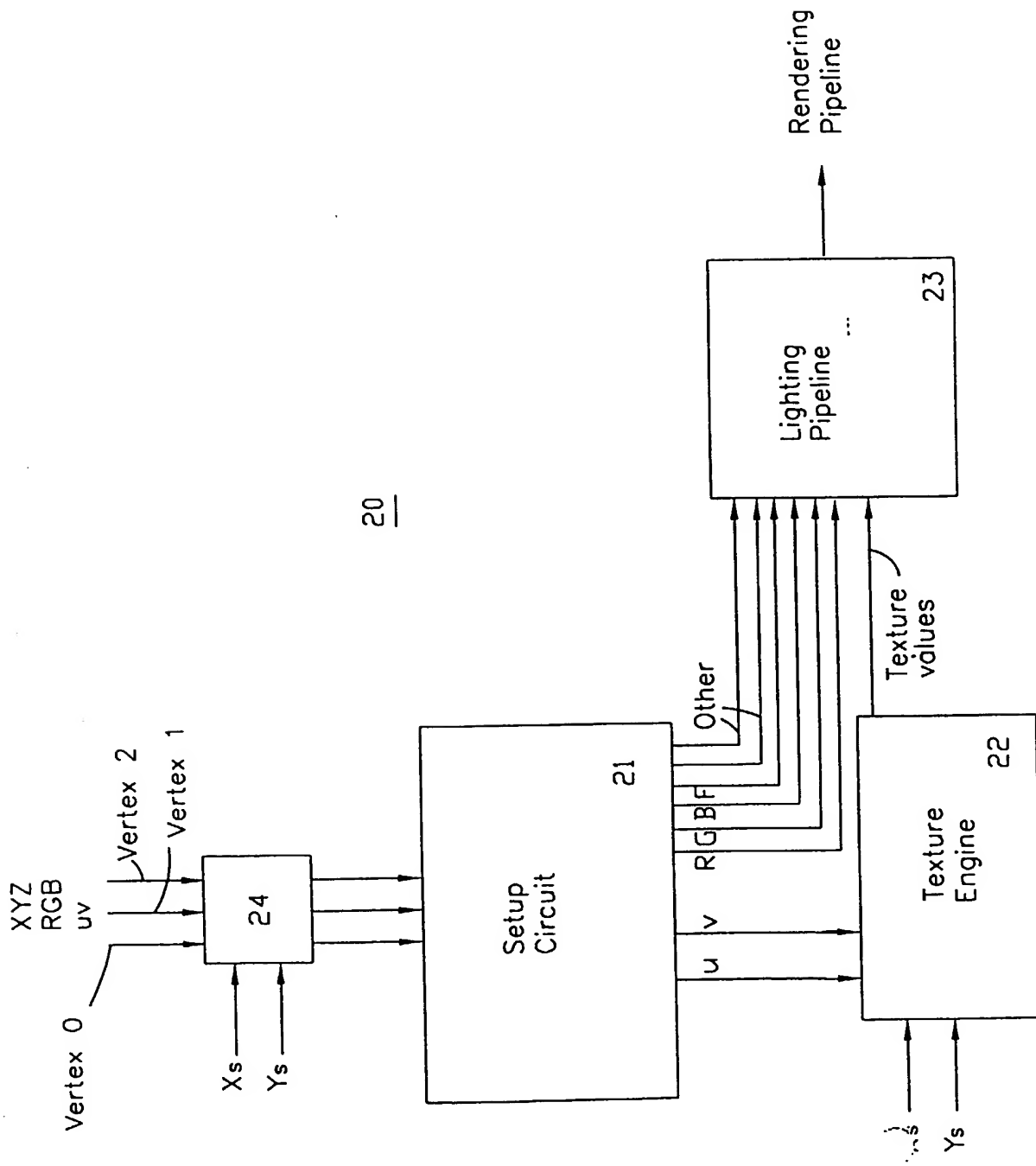
Figure 1

Figure 2 A

Figure 2B

A	x, y, z, r, g, b, u, v, f, ...
B	x, y, z, r, g, b, u, v, f, ...
C	x, y, z, r, g, b, u, v, f, ...
D	x, y, z, r, g, b, u, v, f, ...
E	x, y, z, r, g, b, u, v, f, ...
F	x, y, z, r, g, b, u, v, f, ...
G	x, y, z, r, g, b, u, v, f, ...
H	x, y, z, r, g, b, u, v, f, ...

20

p	x, y, z, r, g, b, u, v, f, ...

Figure 2B

Figure 3

Vertex A	x	y	z	rgb	u	v	f'
Vertex B	x	y	z	rgb	u	v	f
Vertex J	x	y	z	rgb	u	v	f

Command | ABJ |

Figure 3

Figure 4

f/com	X	Y	Z	rgb	U	V

Figure 4
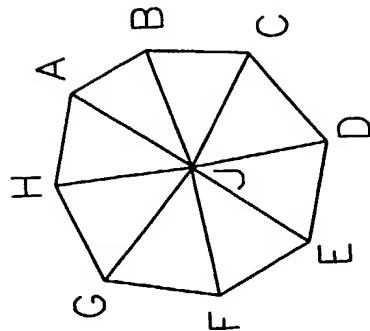
Figure 5

com	X	Y	Z	rgb	U	V	...	U	V	f

Figure 5

Figure 6
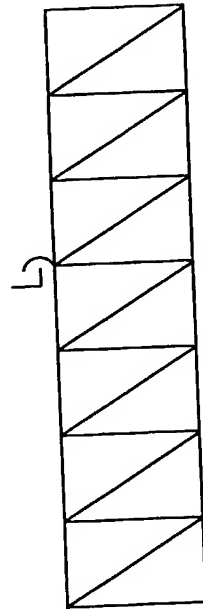
Figure 7

Figure 8

Data replace SM
Controller

Cache 20

Vertex

Tag

test

write

DMA 17

read

Array

Memory
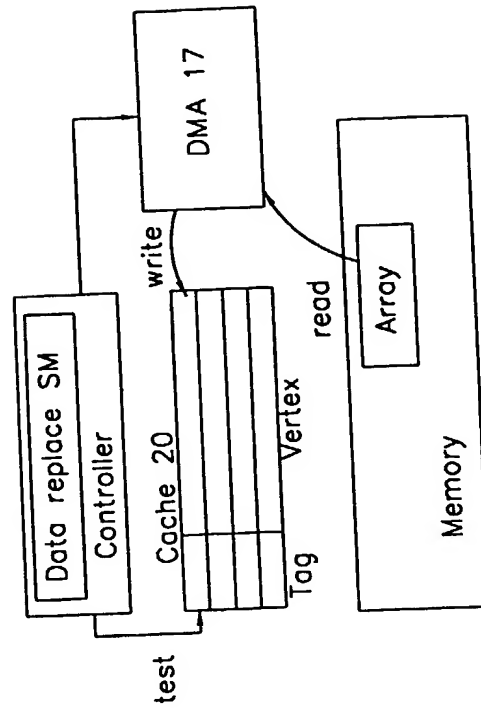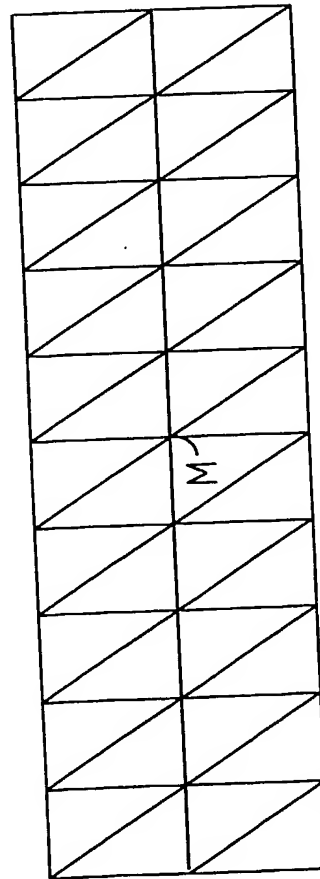
Figure 9

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below, next to my name.

I believe I am the original, first, and sole inventor (if only one name is listed below) or an original, first, and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

METHOD AND APPARATUS FOR PROVIDING A VERTEX CACHE

the specification of which was filed as Serial Number 09/056,386 on April 7, 1998.

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claim(s), as amended by any amendment referred to above. I do not know and do not believe that the claimed invention was ever known or used in the United States of America before my invention thereof, or patented or described in any printed publication in any country before my invention thereof or more than one year prior to this application, that the same was not in public use or on sale in the United States of America more than one year prior to this application, and that the invention has not been patented or made the subject of an inventor's certificate issued before the date of this application in any country foreign to the United States of America on an application filed by me or my legal representatives or assigns more than twelve months (for a utility patent application) or six months (for a design patent application) prior to this application.

I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56.

I hereby appoint Stephen L. King, Reg. No. 19,180; with offices located at 30 Sweetbay Road, Rancho Palos Verdes, California 90275, telephone (310) 377-5073, with full power of substitution and revocation, to prosecute this application and to transact all business in the Patent and Trademark Office connected herewith.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and

1

the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Sole/First Inventor <u>Curtis Priem</u>

Inventor's Signature _____ Date _____5-29-98_____

Residence   <u>Fremont, California</u>   Citizenship   <u>U.S.A.</u>
                (City, State)   (Country)

Post Office Address <u>4052 Kettering Terrace</u>
                      <u>Fremont, California 94536</u>

Full Name of Second/Joint Inventor <u>David Kirk</u>

Inventor's Signature _____ Date __5/29/98__

Residence   <u>San Francisco, California</u>   Citizenship   <u>U.S.A.</u>
                (City, State)   (Country)

Post Office Address <u>2965 Broderick Street</u>
                      <u>San Francisco, California 94123</u>

Applicant:	Priem et al.	Our File:	18659-15C1
Serial No.:	09/056,386	Group Art Unit:	2776
Filed:	April 7, 1998	Examiner:	Tung, K.
For:	**"METHOD AND APPARATUS FOR PROVIDING A VERTEX CACHE"**		

CHANGE OF ADDRESS OF ATTORNEY OF RECORD

RECEIVED

Assistant Commissioner for Patents
Washington, D.C. 20231

MAY 3 1 2002

OPLA

Dear Sir:

Please address all further correspondence in the above-identified application to:

Leah Sherry
OPPENHEIMER WOLFF & DONNELLY LLP
1400 Page Mill Road
Palo Alto, CA 94304

CUSTOMER NO. 25696

Respectfully submitted,

Date: October 25, 2000

Leah Sherry
Reg. No. 43,918

OPPENHEIMER WOLFF & DONNELLY LLP
Customer No. 25696
1400 Page Mill Road
Palo Alto, CA 94304
Telephone: (650) 320-4000
Facsimile: (650) 320-4100